

**PATENT**  
**IBM Docket No. AUS9-2001-0482**

**Amendments to the Claims:**

**Please amend claims 1, 6, 11, 16, 19, 20, 21 and 23 as follows:**

**Please add new claim 24.**

1. (Currently amended) A method of operating a data processing system, having a microprocessor that executes program instructions, comprising the steps of:  
determining an order of said program instructions;  
moving a ~~certain one of said program instructions~~ load instruction to an advanced position in said program order;  
speculatively executing said load instruction;  
checking status information associated with said ~~certain program~~ load instruction to determine whether it will execute successfully; and  
retrying said ~~certain program~~ load instruction by speculatively re-executing said load instruction when said ~~certain program~~ load instruction does not execute successfully;  
wherein said retried load instruction does not call any recovery routines.
2. (Original) A method according to claim 1 wherein said certain program instruction is a load instruction for retrieving information from memory.
3. (Original) A method according to claim 2 wherein said step of checking status information comprises the step of determining whether the information to be retrieved by said load instruction is valid based on the state of a status bit.
4. (Original) A method according to claim 3 wherein said step of retrying comprises the step of loading, based on said status bit, the data again without implementing any recovery procedures.

**PATENT****IBM Docket No. AUS9-2001-0482**

5. (Original) A method according to claim 4 wherein said step of checking is implemented by a speculative load check program instruction.
6. (Currently amended) A data processing system, having a microprocessor that executes program instructions, comprising:
- means for determining an order of said program instructions, and for moving a ~~certain one of said program instructions~~ load instruction to an advanced position in said program order;
  - means for speculatively executing said load instruction;
  - means for checking status information associated with said ~~certain program~~ load instruction to determine whether it will execute successfully; and
  - means for retrying said ~~certain program~~ load instruction by speculatively re-executing said load instruction when said ~~certain program~~ load instruction does not execute successfully, wherein said retried load instruction does not call any recovery routines.
7. (Original) A system according to claim 6 wherein said certain program instruction is a load instruction for retrieving information from memory.
8. (Original) A system according to claim 7 wherein said means for checking comprises means for determining whether the information to be retrieved by the load instruction is valid, based on the state of a status bit.
9. (Original) A system according to claim 8 wherein said means for retrying comprises means for loading, based on the status bit, the data again without implementing any recovery procedures.
10. (Original) A system according to claim 9 wherein said means for checking is implemented by a speculative load check program instruction.

**PATENT****IBM Docket No. AUS9-2001-0482**

11. (Currently amended) A computer program product, including a plurality of program instructions which are executable by a microprocessor in a data processing system and stored on a computer readable medium, said computer program product, comprising:

means for determining an order of said program instructions, and for moving a ~~certain one of said program instructions~~ load instruction to an advanced position in said program order;

means for checking status information associated with said ~~certain program~~ load instruction to determine whether it will execute successfully;

means for speculatively executing said load instruction; and

means for retrying said ~~certain program~~ load instruction by speculatively re-executing said load instruction when said ~~certain program~~ load instruction does not execute successfully;

wherein said retried load instruction does not call any recovery routines.

12. (Original) A computer program product according to claim 11 wherein said certain program instruction is a load instruction for retrieving information from memory.

13. (Original) A computer program product according to claim 12 wherein said means for checking comprises means for determining whether the information to be retrieved by the load instruction is valid, based on the state of a status bit.

14. (Original) A computer program product according to claim 13 wherein said means for retrying comprises means for loading, based on said status bit, the data again without implementing any recovery procedures.

15. (Original) A computer program product according to claim 14 wherein said means for checking is implemented by a speculative load check program instruction.

16. (Currently amended) A method of operating a data processing system, having a microprocessor that executes program instructions, comprising the steps of:

determining an order of said program instructions;

**PATENT****IBM Docket No. AUS9-2001-0482**

moving a load instruction ~~from~~ for retrieving information from memory to an advanced position in said program order;

checking status information associated with said load instruction to determine if the information being retrieved is valid, based on the state of a status bit;

speculatively executing said load instruction; and

retrying speculative execution of said ~~certain program~~ speculatively executed load instruction when said ~~certain program instruction does not execute successfully its initial execution is~~ unsuccessful;

wherein said retried speculative load instruction does not call any recovery routines.

17. (Original) A method according to claim 16 wherein said step of checking status information determines whether the information to be retrieved by said load instruction is valid based only on the state of a status bit.

18. (Original) A method according to claim 17 wherein said step of retrying comprises the step of loading, based on said status bit, the data again without implementing any recovery procedures.

19. (Currently amended) A data processing system, having a microprocessor that executes program instructions including a load instruction which retrieves information from a storage location, comprising:

a compiler that determines an order of said program instructions, and is capable of moving said load instruction to an advanced position in said program order;

a status register that stores an indication of the validity of said load instruction; ~~and~~

a control unit that processes a check load instruction and determines the state of said status register;

an execution unit that speculatively executes said load instruction; and

wherein speculative execution of said load instruction is retried based on the state of said status register; and

wherein said retried speculative load instruction does not call any recovery routines.

**PATENT**  
**IBM Docket No. AUS9-2001-0482**

20. (Currently amended) A system according to claim 19 wherein execution is retried based solely on the state of said indication in said status-bit register and said information is loaded again without implementing any recovery procedures.

21. (Currently amended) A data processing system, comprising

a memory;

a microprocessor that executes program instructions including a load instruction which retrieves information from said memory;

a compiler that determines an order of said program instructions, and is capable of moving said load instruction to an advanced position in said program order;

a status register that stores an indication of the validity of said load instruction; and

a control unit that processes a check load instruction to determine the state of said status register;

an execution unit in said microprocessor that speculatively executes said load instruction; and

wherein speculative execution of said load instruction is retried based on the state of said status register; and

wherein said retried speculative load instruction does not call any recovery routines..

22. (Original) A system according to claim 21 wherein said status register is a single bit and said execution is retried when said bit is set.

23. (Currently amended) A data processing system, comprising

a memory;

a microprocessor that executes program instructions including a load instruction which retrieves information from said memory;

a compiler that determines an order of said program instructions, and is capable of advancing said load instruction to an advanced position in said program order;

a status register that stores an indication of the validity of said load instruction; and

**PATENT****IBM Docket No. AUS9-2001-0482**

a control unit that processes a check load instruction to determine the state of said status register;

an execution unit in said microprocessor that speculatively executes said load instruction; and  
wherein speculative execution of said load instruction is retried based on the state of said status register and independent of other status information; and  
wherein said retried speculative load instruction does not call any recovery routines.

24. (New) A data processing system, comprising

a memory;

a microprocessor that executes program instructions including a load instruction which retrieves information from said memory;

a compiler that determines an order of said program instructions, and is capable of advancing said load instruction to an advanced position in said program order;

a load address table for storing an address of said load instruction to be speculatively executed by said microprocessor; and

an execution unit in said microprocessor that speculatively executes said load instruction, and determines if said load instruction executed successfully by executing an instruction to check whether said address of said load instruction remains stored in said load address table;

wherein speculative execution of said load instruction is retried based on the presence of said address of the speculatively executed load instruction in said load address table; and

wherein said retried speculative load instruction does not call any recovery routines.